Tampereen yliopisto

Josefiina Virtanen

DE BRUIJN GRAAFIT JA NIIDEN KÄYTTÖ DE NOVO GENOMIN KASAAMISESSA

Kandidaattitutkielma

Informaatioteknologian ja viestinnän tiedekunta
Kandidaattitutkielma
Maaliskuu 2020

TIIVISTELMÄ

Josefiina Virtanen: De Bruijn graafit ja niiden käyttö de novo genomin kasaamisessa
Kandidaattitutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Maaliskuu 2020

Tutkielma avaa sitä, kuinka de Bruijn graafeihin ja Eulerin polkuun perustuvia menetelmiä voidaan sekä teoriassa että käytännössä hyödyntää tuntemattoman genomin kasaamisessa. Tavoitteena on esittää selkeä ja tiivistetty määritelmä de Bruijn graafista, esitellä bioinformatiikan sovellusalue, jonka ongelmiin graafiteorialla etsitään vastauksia ja avata de Bruijn graafin rakennetta ja osuutta genomia kasaavissa ohjelmistoissa.

Avainsanat: De Bruijn graafit, graafialgoritmit, bioinformatiikka

1. Johdanto

Eliöiden perimän selvittäminen on ollut biotieteiden keskeisimpiä kysymyksiä jo vuosikymmenet. Ihmisen genomin ensimmäinen versio valmistui 2000-luvun alussa, se oli kolmetoista vuotta vaatinut, usean eri tieteenalan yhteistyönä tehty projekti, joka maksoi kolme miljardia dollaria [Vargas 2002]. Nykyään, pian 20 vuotta myöhemmin, samaan pystytään muutamassa päivässä. Sekvensointitekniikat ovat halventuneet ja ihmisen genomin saa sekvensoitua tuhannella eurolla. Uudet, halvemmat menetelmät tuottavat lyhyempiä jaksoja dataa, mikä ei ole ongelma kun genomi kasataan valmiin mallin perusteella mutta lisää omat haasteensa tiedon käsittelyyn kun referenssiä ei ole saatavilla. Tässä tutkielmassa käsitellään *de novo* genomin kasaamista eli eliön aiemmin tuntemattoman perimän selvittämistä. Genomin kasaaminen ilman mallia antaa lisää tietoa geneettisistä muutoksista ja sairauksista ja uusien lajien perimän selvittäminen valottaa evoluutiobiologian kysymyksiä eliöiden kehityksestä ja sukulaisuudesta.

De Bruijn graafi on merkkijonoja säilövä ja järjestelevä tietorakenne. Ominaisuuksiensa vuoksi de Bruijn graafi sopii ratkaisuksi tietynlaisiin monimutkaisiin laskennallisiin ongelmiin silloin, kun käsiteltävä data voidaan esittää joukkona merkkijonoja. Genomin kasaaminen ilman mallia on laskennallisesti raskas prosessi datan monimutkaisuuden ja suuren määrän takia eikä siihen vielä tunneta jokaisessa tilanteessa täydellisesti toimivaa ratkaisua. Lyhyillä jaksoilla sekvenssidataa de Bruijn graafin pohjalta rakennettu ohjelmisto toimii kuitenkin hyvin.

Tässä tutkielmassa esitellään ensin de Bruijn graafi tietorakenteena ja bioinformatiikan sovellusalue, jolla sitä hyödynnetään. Lopuksi paneudutaan sekä teoreettisella että konkreettisella tasolla tapoihin, joilla de Bruijn graafiin perustuvia algoritmeja on valjastettu ratkaisemaan de novo genomin kasaaminen. Tutkielman toisessa luvussa määritellään de Bruijn graafi tietorakenteena ja sen sisältämä kierros ja polut. Näiden yhteydessä puhutaan de Bruijn graafin säilömän tiedon muodosta ja erityispiirteistä, tavasta jakaa tieto de Bruijn graafin solmuihin ja liittää solmuja toisiinsa kaarilla sekä lopulta keinoista hakea tieto de Bruijn graafista järjestetyssä muodossa. Kolmannessa luvussa kerrotaan de novo genomin kasaamisesta ja puhutaan pääpiirteittäin biologisesta datasta, jonka järjestelyn ongelmaan tutkielmassa esitetyillä menetelmillä on pyritty löytämään ratkaisu. Tutkielman ollessa tietojenkäsittelytieteellinen ei biologiaan syvennytä kovinkaan tarkasti, sen sijaan pyritään antamaan pääpiirteittäinen käsitys biologisen datan erityispiirteistä siltä osin, kun se on oleellista algoritmien toiminnan ymmärtämisen ja perustelun kannalta. Neljännessä luvussa sovelletaan de Bruijn graafeihin ja Eulerin polkuun perustuvaa ratkaisua de novo genomin kasaamiseen ja perustellaan, miksi kyseinen ratkaisu on toimiva. Lopuksi viidennessä luvussa kuvataan

yksityiskohtaisemmalla tasolla de Bruijn graafin rakennetta ja toimintaa genomia kasaavissa ohjelmistoissa. Viimeinen luku sisältää yhteenvedon. Tutkielman teoreettinen osuus perustuu tieteellisistä julkaisuista koottuun tietoon ja käytännön osuus ohjelmistojen virallisiin kuvauksiin.

2. De Bruijn graafi

De Bruijn graafin on virallisesti kehittänyt hollantilainen matemaatikko Nicolaas Govert de Bruijn ja se julkaistiin ensimmäisen kerran hänen teoksessaan "A Combinatorial Problem" vuonna 1946. De Bruijn graafi on solmuista ja niiden välisistä kaarista koostuva verkkomainen tietorakenne. Sen yksilöivät piirteet muihin graafeihin verrattuna liittyvät solmujen säilömään tietoon ja sääntöön, jonka perusteella solmut yhdistetään kaarilla. Kulkemalla de Bruijn graafin läpi solmusta toiseen kaarien osoittamaa reittiä saadaan niiden sisältämästä tiedosta koostettua loogisesti jatkuva sekvenssi. Ominaisuuksiensa vuoksi de Bruijn graafi on käytännöllinen osa bioinformatiikan ongelmien ratkaisua, lisäksi sillä on useita muita sovellusalueita mm. pseudosatunnaisnumerogeneraattorit ja kryptografian menetelmät.

2.1 Yleinen määrittely

Yleisen määrittelyn [Moreno 2005] mukaan de Bruijn graafi on suunnattu graafi, joka esittää valitusta äärellisestä symbolijoukosta S muodostettujen samanmittaisten merkkijonojen limittyvyyttä. Graafi muodostetaan merkkijonojen pohjalta. Merkkijonojen joukko D muodostuu symbolijoukon S alkioiden järjestyksistä siten, että jokainen merkkijono esiintyy joukossa D korkeintaan kerran. Kun yksittäisen joukkoon D kuuluvan merkkijonon todellinen pituus on k, on de Bruijn graafin G sisältämään solmuun u säilötty merkkijono pituudeltaan k-1.

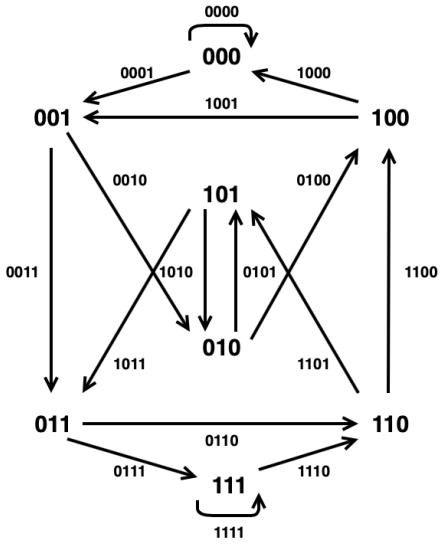
Solmujoukko V graafille G, jolla on k mittaisia merkkijonoja säilövä joukko D, määritellään:

$$V(G^D) = \{\, u \in S^{k-1} \mid u \text{ on alku- tai loppuosa jossain merkkijonossa, joka kuuluu joukkoon } D \,\}.$$

Tieto merkkijonon viimeisestä alkiosta säilötään kaareen. Suunnattu kaari solmusta toiseen on olemassa jos ja vain jos solmut muodostavat yhdessä joukkoon D kuuluvan merkkijonon w, jossa alku on kaaren lähtösolmussa ja loppu solmussa, johon kaari suuntautuu. Toisin sanoen kaari $\alpha v,\ v\beta$ on siis olemassa, jos sekä α että β kuuluvat symbolijoukkoon S ja $\alpha v\beta$ kuuluu merkkijonojoukkoon D. Kaarijoukko E määritellään:

$$E(G^D) = \{ (\alpha v, v\beta) \mid \alpha,\beta \in S, \alpha v\beta \in D \}.$$

Tämä tarkoittaa käytännössä sitä, että merkkijono $\alpha v\beta = w$ on pituudeltaan k, osa αv kattaa sen alkiot järjestyksessä ensimmäisestä indeksistä toiseksi viimeiseen indeksiin pituudelta k-1 ja osa $v\beta$ toisesta indeksistä viimeiseen indeksiin yhtälailla pituudelta k-1. Sitä osaa v merkkijonossa w, joka esiintyy sekä merkkijonoissa αv että $v\beta$, kutsutaan merkkijonon w tekijäksi. Merkkijonossa αv se on loppuosa ja merkkijonossa $v\beta$ alkuosa [Moreno 2005] . Kuvassa 1. on esitetty de Bruijn graafi, jossa symbolijoukko S={0,1} ja k=4. Kaaret kulkevat Eulerin kierroksen.



Kuva 1. *De Bruijn graafi.*

De Bruijn graafilla on joitakin erikoistapauksia: ensiksi, solmusta v voi suuntautua kaari siihen itseensä, mikäli sen merkkijono koostuu vain yhdestä symbolista. Toiseksi, mikäli solmuihin talletettujen merkkijonojen pituus on 1, ovat kaikki solmut yhdistetty kaarilla toisiinsa. Kolmanneksi, jos joukko D koostuu kaikista mahdollisista tavoista järjestää symbolijoukko S pituudella k, on kaikkien solmujen välillä sama määrä kaaria.

2.2 Eulerin kierroksen ja polun piirteitä

Eulerin graafi on graafi, joka sisältää Eulerin *kierroksen* (engl. cycle, circuit). Eulerin kierros on graafin läpi kulkeva reitti, joka kulkee kerran jokaisen kaaren kautta ja päätyy lopuksi samaan solmuun, josta kierros alkoi. Eulerin kierros on olemassa kun jokaisesta solmusta voi kulkea kaarien kautta mihin tahansa toiseen solmuun eli graafi on *kytketty* (engl. connected) ja jokaisen solmun ulko- ja sisäaste ovat yhtä suuret eli jokaisesta solmusta lähtee kaaria yhtä paljon kuin siihen suuntautuu niitä. Eulerin *polku* (engl. path, trail) kulkee vastaavasti jokaisen kaaren läpi mutta sen alku- ja loppusolmut

eivät ole samat. Eulerin polku voidaan kulkea mikäli graafin kahden solmun asteet ovat parittomat: toisen solmun ulkoasteen erotus sisäasteesta on yksi ja toisen solmun sisäasteen erotus ulkoasteesta on yksi. Tällöin nämä kaksi solmua ovat polun alku ja loppu.

De Bruijn sekvenssi saadaan kulkemalla Eulerin kierros graafin läpi. De Bruijn sekvenssi *B* merkkijonojoukolle *D* on pituudeltaan |*D*| ja se sisältää jokaisen joukkoon *D* kuuluvan merkkijonon sekvenssin tekijänä [Moreno 2005]. Toisaalta geneettisen datan ominaisuudet osaltaan poissulkevat mahdollisuuden Eulerin kierroksesta ja de Bruijn sekvenssistä. Vaikka bakteereilla ja arkeilla genomi voi olla syklinen rengas, tumallisten eli eukaryoottien perimä koostuu kromosomeista, joilla on alku ja loppu, eikä niiden odoteta limittyvän. Tässä tapauksessa geneettisen datan pohjalta muodostetun graafin käsittelyssä pyritään löytämään kunkin kromosomin kattava Eulerin polku kierroksen sijasta. [Compeau *et al.* 2011]

3. De novo genomin kasaaminen

Termi *de novo* on latinaa ja sen merkitys suomeksi on "uusi" tai "alusta alkava". De novo genomin kasaaminen tarkoittaa laskennallista tapaa koota genomi alkuperäiseen järjestykseen sekvensoimalla saatujen lyhyiden *jaksojen* (engl. read) perusteella. Genomi koostuu eliön perinnöllisen tiedon sisältämästä deoksiribonukleiinihaposta eli DNA:sta ja DNA osaltaan neljää erilaista emästä toistavista kaksoiskierteisistä ketjuista. Jokainen erilaisen molekyylirakenteen omaava emäs koodaa niin sanotusti omaa kirjaintaan A, C, G tai T ja kaksoiskierteessä ne liittyvät toisiinsa pareina A-T ja C-G. Termi Next Generation Sequencing (NGS) kattaa erilaisia menetelmiä selvittää DNA:n emäsjärjestys ja kääntää se luettavaan muotoon. Sekvensoinnin aikana DNA hajoaa lyhyempiin osiin ja tuloksena on erimittaisia jaksoja dataa sekoittuneessa järjestyksessä. Jaksot kattavat osia DNA-ketjuista kohtuullisen epätasaisesti, ne voivat olla päällekkäisiä tai jättää osia välistä.

Sekvensointimenetelmistä valtavirtaa on Illumina, joka tuottaa noin 50-250 emäsparin mittaisia jaksoja sekvenssidataa. Myöhemmin esiteltävät ohjelmistot on kirjoitettu erityisesti lyhyille, Illumina-tyylisille jaksoille. Menetelmän toiminta on hyvin lyhyesti ja yksinkertaistetusti seuraava: fragmentit kaksoiskierteistä DNA:ta erotetaan toisistaan ja niiden päihin lisätään osat, joilla ne kiinnittyvät alustaan laitteessa. Sen jälkeen jokaista fragmenttia monistetaan sen alkuperäisen tarttumakohdan ympärille, jotta muodostuu klustereita, joissa on sama sekvenssi samoin päin. Lähetetään yksi kerrallaan fluoresoituja emäksiä ja niiden tarttuessa erittyvän aallonpituuden perusteella kerätään talteen tieto niiden järjestyksestä kussakin klusterissa. Tämän jälkeen

muodostetaan fragmenteista kopiot, jotka kiinnittyvät ylösalaisin ja lähetetään taas fluoresoituja emäksiä, jotta saadaan luettua kaksoiskierteisen DNA:n toinenkin puoli. Prosessi käsittelee samanaikaisesti miljoonia rinnakkaisia fragmentteja ja ihmisen genomin sekvensointi kestää noin päivän. [Bentley *et al.* 2008]

De novo genomin kasaamisessa pyritään siis näin kerätyn datan perusteella selvittämään mahdollisimman tarkkaan, missä järjestyksessä sekvensoitu DNA oli luonnossa eli solun sisällä kromosomeissa. Tätä ongelmaa ratkaistaan de Bruijn graafeihin perustuvilla algoritmeilla.

Koska de novo -lähestymistapa ei vaadi aiempaa tietoa genomin järjestyksestä, sitä hyödynnetään muun muassa silloin kun referenssiä ei ole tai se on huonolaatuinen eli saattaa sisältää aukkoja tai olla fragmentoitunut. Referenssisekvenssillä tarkoitetaan aiemmin kasattua genomia, joka on yleensä mosaiikki useamman saman lajin yksilön perimästä ja jota voidaan käyttää mallina muiden kyseisen lajin edustajien genomin kasaamisessa. Ihmiselle on olemassa useampia referenssejä ja niistä pyritään jatkuvasti saamaan kattavampia versioita.

Jos referenssi on olemassa on usein käytännöllisempää hyödyntää *linjausta* (engl. genome alignment). Tämä tarkoittaa yksinkertaisesti sitä, että sekvensoimalla saatuja DNA-jaksoja verrataan referenssiin ja ne asetetaan järjestykseen sen perusteella. Näin toimitaan pääsääntöisesti ihmisen genomin kanssa: linjaus on nopeampaa ja kuluttaa vähemmän resursseja. Lisäksi linjausalgoritmit pystyvät ottamaan huomioon myös lyhyempiä jaksoja sekvenssidataa, jolloin saadaan parempi kuva datan kattavuudesta, mikä on oleellista esimerkiksi mitatessa tiettyjen geenien aktivoitumista eri olosuhteissa.

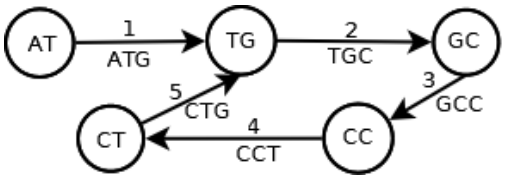
4. Graafiteoria ratkaisuna genomin kasaamisessa

Suuri harppaus genomia kasaavien algoritmien kehityksessä oli ajatus siitä, että ongelma voidaan esittää de Bruijn graafin avulla variaatiolla Eulerin polusta aiemmin käytetyn Hamiltonin polkuun perustuvan lähestymistavan sijaan. Eulerin polkuun perustuvan ratkaisun hyödyntäminen on huomattavasti käytännöllisempää, koska se on laskennallisesti kevyempi verrattuna Hamiltonin polkuun ja voi tuottaa silti saman sekvenssin. Hamiltonin polkua vastaava lopputulos saadaan polynomisessa ajassa etsimällä Eulerin polku graafin version läpi kun taas Hamiltonin polun etsintä tapahtuu yleisesti eksponentiaalisessa ajassa, se on NP-täydellinen ongelma eikä siihen suurilla datamäärillä ole tehokasta ratkaisua [Kozak *et al.* 2013]. Nykyään yleisesti käytetyt, halvemmat ja nopeammat NGS-tekniikat tuottavat kymmenistä miljoonista yli sataan

miljoonaan lyhyttä, rinnakkaista jaksoa, riippuen genomin koosta ja valitusta jaksojen pituudesta. Jaettuina sellaisenaan niiden limittymistä kuvaavan graafin solmuihin, ne kasvattaisivat graafin liian suureksi. Pienten mikrobigenomien ja ihmisen genomin ensimmäisen, 2000-luvun alussa valmistuneen version (Human Genome Project) tuottamiseen on hyödynnetty Hamiltonin polkua [Compeau *et al.* 2011] mutta on hyvä ottaa huomioon muun muassa, että kyseinen projekti kesti 13 vuotta ja silloin käytössä ollut Sanger-sekvensointi tuotti pienemmän määrän pidempiä, yli 500 emäsparin mittaisia jaksoja. Nykyään de Bruijn graafeihin perustuvia ohjelmistoja pidetään parhaina de novo kasaamisessa [Khan *et al.* 2018].

4.1 Periaate Eulerin polkuun perustuvan algoritmin taustalla

Ensimmäinen de Bruijn graafin ja Eulerin polun pohjalta kehitetty genomia kasaava ohjelmisto oli EULER [Pevzner *et al.* 2001]. Se loi pohjan muille myöhemmin kehitettäville Eulerin polkuun ja de Bruijn graafeihin perustuville ohjelmistoille. Yleinen ajatus de Bruijn graafin käytön taustalla on seuraava [Compeau *et al.* 2011]: Jaksot jaetaan lyhyisiin, erillisiin *k*:n mittaisiin osiin, joista käytetään tieteellisessä tekstissä usein nimitystä *k-mer* perustuen vakioon *k* ja kreikan kielen sanaan *mer*, joka tarkoittaa osaa. Useimmiten *k* on pariton luku väliltä 21..27. De Bruijn graafin yleisen määrittelyn yhteydessä kuvattu symbolijoukko *S* koostuu emäksiä esittävistä merkeistä {A, C, G, T}, lisäksi tuntematonta emästä kuvataan valitulla merkillä, joka voi olla esimerkiksi N. Jaksojen jakaminen tapahtuu leikkaamalla jakson jokaisen alkavan indeksin kohdalta *k*:n mittainen merkkijono kunnes päästään jakson loppuun. Kun jakson pituus on *n*, on siitä saatavien *k*-merien määrä *n*-*k*+1. Havainnollistaen ongelmaa hyvin lyhyellä ja yksinkertaistetulla esimerkillä, valitaan jakso ATGCCTG. Leikattuna se loisi merkkijonojoukon *D* = {ATG, TGC, GCC, CCT, CTG} kun *k*=3. Koska de Bruijn graafin solmuihin talletettavat merkkijonot ovat pituudeltaan *k*-1 ja solmujoukon *V* tulee sisältää kerran jokainen joukon *D* alkion alku- ja loppuosa, solmuihin asetetaan *V*={AT, TG, GC, CC, CT}. Suunnattu kaari e_1 muodostetaan solmusta (AT) solmuun (TG), koska ne yhdessä muodostavat joukkoon *D* kuuluvan merkkijonon ATG. Kaari e_1 sisältää tiedon näiden solmujen merkeistä. Muiden kaarien kanssa toimitaan samoin. Kaarijoukko *E* voidaan merkitä {ATG, TGC, GCC, CCT, CTG}. Kuvassa 2. on esitetty kaavio näin muodostetusta de Bruijn graafista.



Kuva 2. *Yksinkertaistettu de Bruijn graafi. Numeroidut kaaret kulkevat Eulerin polun.*

Sekvenssi saadaan kulkemalla Eulerin polku muodostetun de Bruijn graafin läpi. Huomataan, että solmu (AT) on polun alku, koska sen sisäaste on ulkoastetta pienempi. Valitaan AT sekvenssin aluksi. Kuljetaan järjestyksessä jokaisen kaaren läpi ja lisätään sekvenssiin samalla jokaisen kaaren osoittama merkki, jolla edellisen solmun sisältämä merkkijono jatkuu, kunnes päädytään viimeiseen solmuun eikä kulkemattomia kaaria enää ole. Näin ollaan muodostettu de Bruijn graafin pohjalta alkuperäistä jaksoa kuvaava sekvenssi ATGCCTG. Usean erillisen jakson kanssa toimittaisi täysin samalla periaatteella, jokainen jakso leikattaisi *k*:n mittaisiin osiin ja uniikit osat jaettaisi edelleen solmuihin. Olettaen, että jaksot limittyvät siististi, saataisi niistä koostettua yhtenäinen sekvenssi.

Virheettömällä ja tasaisen kattavuuden omaavalla sekvenssidatalla muodostettu de Bruijn graafi sisältää aina Eulerin polkuja. Toisaalta polku graafin läpi ei aina ole yksikäsitteinen: geneettinen koodi itsessään sisältää sen käsittelyä vaikeuttavia piirteitä, kuten toistot, matalan kompleksisuuden osat ja vain samaa symbolia sisältävät osat. Kun sama merkkijono tai samojen merkkijonojen muodostama järjestys esiintyy eri kohdissa genomia ja sen alku- tai loppuosan säilömästä solmusta lähtee ja siihen suuntautuu useampia kaaria, sama graafi voidaan kulkea eri järjestyksissä riippuen risteyskohdissa valituista suunnista. Tällöin tarvitaan tietoa, jonka perusteella päättää, mikä lopputuloksista on oikea. Sekvensoinnin aikana tapahtuneet virheet ja datan epätäydellinen kattavuus vaikuttavat myös lopputulokseen, ne voivat luoda vääriä solmuja ja vääriä kaaria graafiin, mikä aiheuttaa umpikujaan päättyviä *haaroja* (engl. branch) ja *kuplia* (engl. bubble). Nämä hankaloittavat oikean polun löytämistä. Koska kasaamisvaiheessa ei vielä lähtökohtaisesti ole oikeaa käsitystä siitä, millaisia sekvenssejä tuntemattoman genomin voidaan olettaa sisältävän, on algoritmilla rajallisesti tietoa, jonka perusteella erottaa virheettömät osat virheellisistä. Virheiden korjaamiseksi on kuitenkin erilaisia menetelmiä, haarat voidaan leikata, kuplia voidaan puhkaista ja toistojen aiheuttama rinnakkaisten kaarien ongelma voidaan ratkaista tarkastamalla toistojen oikea järjestys leikkaamattomista jaksoista, mikäli ne kattavat toistoja sisältävät osat riittävältä pituudelta. [Pevzner *et al.* 2001]

De novo kasaaminen vaatii paljon aikaa ja muistia. Ihmisen genomi on yli 3 miljardin emäsparin mittainen ja se on säilöttynä 23 pariin kromosomeja, joiden pituudet vaihtelevat 50-300 miljoonan välillä. Ihmisen genomin kasaamisessa keskusmuistin käytön huippu voi ohjelmistosta riippuen olla yli 600 GB ja kasaaminen voi vaatia kymmeniä tunteja aikaa [Jackman *et al.* 2017]. Ohjelmistojen vertailussa [Khan *et al.* 2018] käytetty ihmisen genomi koostui 126 605 856 limittyvästä, korkeintaan 100 emäsparin mittaisesta jaksosta ja genomista riippuen vastaavalla datalla muodostettu de Bruijn graafi sisältää miljoonia solmuja kun *k*-merin pituus on välillä 21..27. Lisäksi

koska jokainen *k*-mer voi esiintyä de Bruijn graafissa vain kerran huolimatta siitä, miten monta kertaa se esiintyy genomissa, solmujen välillä voi olla useita, rinnakkaisia kaaria. Optimoitukin ratkaisu tarvitsee huomattavasti resursseja. [Khan *et al.* 2018]

4.2 Vaihtoehtoisia tapoja

De Bruijn graafien lisäksi käytetään muun muassa *overlap-layout-consensus* (OLC) -lähestymistapaa. Tällöin vaihtelevan mittaiset jaksot säilötään solmuihin leikkaamatta niitä, etsitään limittyvät jaksot solmuista ja yhdistetään ne kaarella. Tällöin etsitään Hamiltonin polku Eulerin polun sijasta eli kuljetaan graafin läpi käymällä kerran jokaisessa solmussa. Eri tietorakenteita hyödyntäviä ohjelmistoja vertailemalla on todettu, että de Bruijn graafeihin perustuvat ohjelmistot ovat nopeampia verrattuna OLC-tekniikkaan perustuviin mutta toisaalta OLC on parempi vaihtoehto pitkille jaksoille sekvenssidataa [Khan *et al.* 2018]. On mahdollista, että sekvensointitekniikoiden kehittyessä ja mahdollistaessa pidempien yhtenäisten jaksojen tuottamisen palataan takaisin OLC-tekniikkaan.

5. De Bruijn graafeihin perustuvia ohjelmistoja

Tässä luvussa tutustutaan tarkemmin erilaisiin de Bruijn graafin implementaatioihin kahdessa eri ohjelmistossa, ABySS ja Velvet. Ne on valittu tarkasteluun tiettyjen perustavanlaatuisten erojensa vuoksi - ABySS toimii hajautetusti klusteriin kuuluvien koneiden välillä ja toisaalta Velvet on alunperin rakennettu toimimaan yhdellä laitteella. ABySS:n de Bruijn graafi noudattaa rakenteeltaan edellisessä luvussa määriteltyä paradigmaa jakaen yhden tietyn mittaisen *k*-merin jokaiseen solmuun ja toisaalta Velvetin de Bruijn graafin solmu voi koostua yhden tai useamman *k*-merin järjestyksestä. Kumpikin ohjelmistoista on julkaistu GNU General Public License -lisenssin alla. Ohjelmistoista ja niiden toiminnasta annetaan yleiskuva mutta tutkielman fokus on pääasiassa niiden sisältämissä de Bruijn graafeissa.

5.1 ABySS

ABySS (Assembly By Short Sequences) [Simpson *et al.* 2009] on julkaistu vuonna 2009. Se on kirjoitettu C++-kielellä erityisesti suuremmille datamäärille ja se oli ensimmäinen ohjelmisto, joka pystyi kasaamaan ihmisen genomin lyhyistä jaksoista ilman mallia. ABySS:n perustana on hajautettu de Bruijn graafi, jossa k-mereilla on säilömispaikasta kertovat hajautusarvot, jotka on laskettu niiden emäsjärjestyksen perusteella. Tietorakenteen hajauttaminen suurelle määrälle koneita kasvatti käytössä olevan muistin määrää ja täten mahdollisti suuremman datamäärän ja edelleen suuremman genomin käsittelyn. Koneiden välisen kommunikoinnin mahdollistaa Standardized Message-passing System (MPI) ABySS:n ensimmäisessä versiossa.

ABySS perustuu sekä edellisessä luvussa EULER:n taustalla oleviin että myöhemmin tässä luvussa Velvetin yhteydessä kuvattaviin ideoihin de Bruijn graafien käytöstä. Virheiden käsittely on myös verrattain samanlaista kuin Velvetin.

Hajautuksen sujuvuuden kannalta tiedon etsimisen ja tallettamisen tulee olla selkeää ja tehokasta. Tämä on ratkaistu määrittelemällä hajautusfunktio, joka laskee talletuspaikan jokaiselle uniikille solmuun talletettavalle *k*-merille sen emäsjärjestyksen perusteella. Emäkset {A, C, G, T} on tässä kohtaa korvattu numeroilla {0, 1, 2, 3} ja kunkin *k*-merin saaman nelikantaisen arvon mukaan laskettu hajautusarvo on aina uniikki eri *k*-mereille. Sekä *k*-merin että sen *vastakkaisen komplementin* (engl. reverse complement) hajautusarvot lasketaan ja ne yhdistetään XOR-operaatiolla, jonka jakojäännöksestä (modulo koneiden määrä) saadaan solmun talletuspaikan indeksi. Vastakkainen komplementti tarkoittaa sitä emäsjärjestystä, joka tietyssä kohdassa kaksoiskierteistä DNA:ta esiintyisi sen vastakkaisella puolella, eli emäkset olisivat parittain A-T ja G-C. Vastakkaisiin komplementteihin liittyvän säännön mukaan halutaan määritellä vakio *k*, parittoman mittainen *k*-mer ei voi olla itsensä vastakkainen komplementti. [Simpson *et al.* 2009]

Jokaiseen solmuun liittyy yleinen tieto kaaresta. Sen sijaan, että solmulla olisi useampia erillisiä siihen osoittavia ja siitä suuntautuvia kaaria, sillä on yksi yleinen kaari, joka sisältää tiedon mahdollisten kaarien olemassaolosta. Kaari koostuu kahdeksasta bitistä, joista kukin esittää yhtä mahdollista emästä, jolla *k*-mer voi jatkua joko eteen tai taakse. De Bruijn graafissa seuraavalla paikalla olevaan solmuun päästään jälleen hajautusfunktion avulla liittämällä *k*-merin eteen tai taakse se emäs, jolla sekvenssin tiedetään kaaren mukaan jatkuvan ja laskemalla näin saadun uuden *k*-merin talletuspaikka. [Simpson *et al.* 2009]

Ajon ensimmäisessä osassa leikataan jaksot osiin valitun luvun *k* mukaisesti ja muodostetaan de Bruijn graafi. Tähän käytetään vain niin sanotusti täydellisiä sekvenssejä: tuntemattoman emäksen sisältävät osat hylätään. Saadut *k*-merit talletetaan hajautustauluun jollekin klusteriin kuuluvista koneista hajautusfunktion mukaisesti. Vaikka hajautustauluun säilötään vain tiettyä uniikkia sekvenssiä esittävä *k*-mer tai sen vastakkainen komplementti, pidetään yllä myös tietoa siitä, kuinka monta kertaa kyseinen *k*-mer on esiintynyt syötteessä. Solmujen muodostamisen jälkeen niiden kaariin talletetaan tieto vierekkäisistä solmuista de Bruijn graafissa. Tämä tehdään laskemalla jokaisen solmun sisältämän *k*-merin kohdalla hajautusarvo uudelle *k*-merille, joka saataisi lisäämällä vuorotellen aiemman eteen tai taakse joku mahdollisista jatkoista. Mikäli näin saatu uusi *k*-mer on olemassa hajautustauluun tallennettuna,

tiedetään näiden kahden limittyvän ja kaareen talletetaan tieto siitä. [Simpson *et al.* 2009]

Ennen ajon toista osaa korjataan syötteessä mahdollisesti esiintyneistä virheistä johtuvat epäselvyydet de Bruijn graafissa. Virheitä ovat edellisessä luvussa mainitut umpikujaan johtavat haarat ja kuplat. Umpikuja on ongelmista yleisin, siinä virheellinen *k*-mer on sekoittunut virheettömien kanssa ja muodostanut haaran, joka loppuu kesken. Haara jäljitetään alkupisteeseen, jossa voidaan olettaa löytyvän ongelmakohdan ja katkaistaan, mikäli sen pituus on lyhyempi kuin valittu *kynnys* (engl. threshold). Toisaalta on otettava huomioon datan mahdollinen huono kattavuus: umpikuja voi syntyä myös virheettömästä *k*-merista, mikäli sen jatkon kohdalla datassa on aukko. Mikäli tällainen haara poistettaisiin, menetettäisi samalla tärkeää tietoa. Kuplissa taas virheellisestä *k*-merista alkanut haara yhdistyy jonnekin. Tällöin pyritään tunnistamaan syy kuplan muodostumisen taustalla, kuplat voivat virheiden lisäksi johtua alleeleista diploidissa genomissa tai luonnollisista toistoista, ja käsittelemään kupla sen ominaisuuksien mukaan. Virheiden käsittelyn jälkeen yksikäsitteisten kaarien yhdistämät solmut liitetään toisiinsa muodostamaan *jatkumoita* (engl. contig). Ajon toisessa osassa jatkumot liitetään toisiinsa hyödyntäen mahdollisesti saatavilla olevaa kaksiloppuista dataa. [Simpson *et al.* 2009]

Vuonna 2017 julkaistiin ABySS:n seuraava versio, ABySS 2.0. Sen perustana on todennäköisyyslaskentaa hyödyntävä tietorakenne, Bloom-filtteri, jonka pohjalta de Bruijn graafi rakennetaan kasaamisen ensimmäisessä vaiheessa. Bloom-filtterin ansiosta muistivaatimukset ovat huomattavasti pienemmät verrattuna aiempaan versioon, mikä mahdollistaa suurten genomien kasaamisen yhdellä laitteella: keskusmuistin käytön huippu ihmisen genomin kasaamisessa oli 34 GB ja aikavaatimus (wall-clock time) 20 tuntia verrattuna aiemman version 418 GB ja 14 tunnin vaatimuksiin. [Jackman *et al.* 2017]

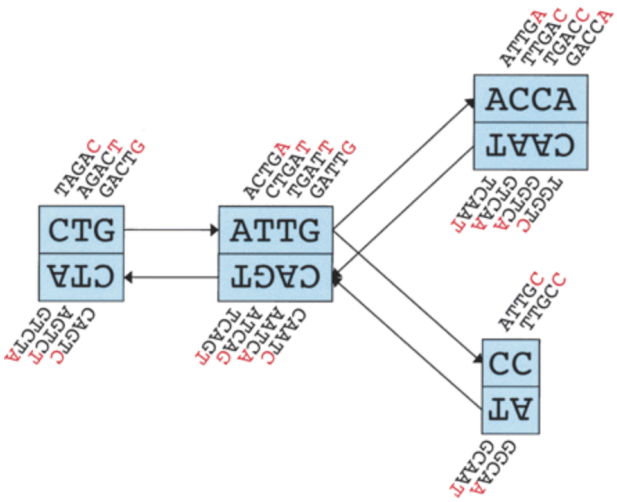
5.3 Velvet

Velvet on genomia kasaava ohjelmisto, jonka ovat kehittäneet Daniel Zerbino ja Ewan Birney Euroopan Bioinformatiikan Instituutissa (EMBL-EBI, Cambridge, UK). Se on pääasiassa suunniteltu lyhyille jaksoille ja pienille genomeille. Velvetin ensimmäinen versio on julkaistu vuonna 2008 ja se on kirjoitettu kokonaan C-kielellä. Velvetin alkuperäisenä tavoitteena oli ratkaista kaksi genomin kasaamiseen läheisesti liittyvää ongelmaa: virheiden käsittely ja DNA-jaksojen luonnolliset toistot. [Zerbino *et al.* 2008]

Sen sijaan, että de Bruijn graafiin talletettaisiin yksittäisiä *k*-mereja tavalla, joka on määritelty tutkielman toisessa ja neljännessä luvussa, Velvetin de Bruijn graafin solmut koostuvat järjestetystä joukosta limittyviä *k*-mereja. Limittyvyys on kuitenkin linjassa aiemmin määritellyn kanssa: solmun aiemman *k*-merin loppuosa on solmun seuraavan *k*-merin alkuosa ja perinteisessä de Bruijn graafissa näiden välille muodostuisi kaari. Solmun sisältämien limittyvien *k*-merien määrää ei ole kiinnitetty ja se muuttuu dynaamisesti ajon aikana. [Zerbino *et al.* 2008]

Jokaisella solmulla on siihen yhdistetty kaksossolmu ja yhdessä näistä käytetään nimitystä *lohko* (block). Kaksossolmun sisältämä tieto on tietynlainen peilikuva, se koostuu oman kaksossolmunsa *k*-merien vastakkaisista komplementeista luettuna vastakkaisesta suunnasta. Koska lohkon tahdotaan pysyvän jatkuvasti yksikäsitteisessä tilassa, solmun muokkaaminen vaatii aina muutosten tekemistä myös kaksossolmussa. [Zerbino *et al.* 2008]

Solmujen ollessa tietyssä mielessä yksinkertaistettuja graafeja sinänsä, on loogista että jokaisella solmulla on sekvenssi. Solmun *u* sekvenssi *s(u)* on sen säilömien *k*-merien viimeisten symbolien muodostama sekvenssi siinä järjestyksessä, kun ne luetaan ensimmäisestä lähtien. Lohkon sisältämien solmujen sekvenssit eivät kuitenkaan välttämättä ole toistensa komplementteja, koska *k*-merit on järjestetty solmuihin peilikuvina: solmun viimeisen *k*-merin viimeinen symboli on sen kaksossolmun ensimmäisen *k*-merin ensimmäisen symbolin pari. Kuvassa 3. solmut ja niiden kaksossolmut on esitetty niiden sekvenssin sisältäminä laatikoina. Solmujen ylä- ja alapuolilla on kuvattu limittyvät *k*-merit, joista sekvenssi muodostuu. [Zerbino *et al.* 2008]



Kuva 3. *Yksinkertaistettu esimerkki Velvetin de Bruijn graafista. [Zerbino et al. 2008]* CC-BY-NC 4.0

Solmujen välille muodostetaan suunnattuja kaaria de Bruijn graafin yleisen määritelmän mukaisesti: kaari solmusta *u* solmuun *v* on olemassa kun solmun *u* viimeisen *k*-merin loppuosa on solmun *v* ensimmäisen *k*-merin alkuosa. Lohkojen välisen yksikäsitteisyyden ylläpitämiseksi muodostetaan kaari myös kaksossolmujen välille: kun kaari suuntautuu solmusta *u* solmuun *v*, tulee myös solmun *v* kaksossolmusta lähteä suunnattu kaari solmun *u* kaksossolmuun. Muistin ja laskutehon säästämiseksi solmuja voidaan yhdistää. Tämä tapahtuu useamman kerran ajon aikana, graafin luomisen ja muokkaamisen, kuten virheiden korjaamisen, yhteydessä. Kun valmiissa de Bruijn graafissa löytyy osia, joissa solmusta *u* lähtee kaari ainoastaan solmuun *v* ja solmuun *v* osoittaa kaari ainoastaan solmusta *u*, solmut *u* ja *v* sekä niiden kaksossolmut yhdistetään toisiinsa uudeksi lohkoksi. [Zerbino *et al.* 2008]

Ajon ensimmäisessä osassa käsitellään syöte muotoon, jonka pohjalta de Bruijn graafin muodostaminen voi tapahtua. Alussa määritellään mitta *k*, jonka mukaan jaksot leikataan. Hajautustauluun talletetaan jokaisen ensimmäisenä löydetyn uniikin *k*-merin sisältäneen jakson ID ja *k*-merin sijainti jakson sisällä. Lisäksi jokaisen *k*-merin ja sen vastakkaisen komplementin välille muodostetaan alusta alkaen yhteys. Ajon toisessa osassa luodaan solmut seuraavaan tietokantaan. Solmu muodostuu järjestetystä joukosta kunkin jakson uniikkeja, limittyviä *k*-mereja siltä matkalta, jolla ne limittyvät aukottomasti jakson sisällä ja kun ne sisältävät vain *k*-mereja, jotka eivät esiinny myöhemmin. Solmujen välille luodaan suunnatut kaaret hyödyntäen *k*-merien muodostamisen yhteydessä saatuja paikkatietoja. Kolmannessa osassa, graafin muodostamisen ja solmujen yhdistämisen jälkeen korjataan sen sisältämät virheet: umpikujat ja kuplat. Umpikujiin johtavat haarat poistetaan graafista, mikäli niiden pituus on pienempi kuin valittu kynnys ja mikäli risteyskohdassa olevasta solmusta haaraan johtaa vähemmän kaaria verrattuna polun toiseen mahdolliseen suuntaan. Kuplien kattamat osat voidaan yhdistää toisiinsa, mikäli ne muodostuvat solmuista, joiden sisältämät *k*-merit ovat riittävän samankaltaista suhteessa tiettyihin vaihteleviin sekvenssiin ja graafiin liittyviin ominaisuuksiin. Lopuksi neljännessä osassa, virheiden korjaamisen jälkeen, ratkaistaan kaksiloppuista dataa hyödyntäen toistot yhdistettyjen solmujen muodostamien jatkumoiden väliltä, jotta voidaan varmistaa niiden oikea järjestys. [Zerbino *et al.* 2008]

Kasaamisen aika- ja tilavaatimukset ovat riippuvaisia genomin koosta. Huomattavasti ihmisen genomia pienemmän, noin kahden miljoonan emäsparin mittaisen S*treptococcus* -bakteerin genomin sisältämä graafi vaatii 2 GB keskusmuistia ja sen kasaaminen onnistuu kolmessa minuutissa. Toisaalta suurempien genomien kohdalla vaatimukset kasvavat hurjasti. [Zerbino *et al.* 2008]

6. Yhteenveto

De Bruijn graafi on yksinkertainen vastaus monitahoiseen ongelmaan. Se on osaltaan helpottanut tuntemattoman genomin kasaamisen ongelmaa mahdollistamalla kevyemmän lähestymistavan siihen. De Bruijn graafin ja Eulerin polun variaatioiden avulla on viime vuosien aikana onnistuttu tuottamaan vertailuissa parhaiten menestyviä ohjelmistoja [Khan *et al.* 2018].

Tutkielmassa käsiteltiin genomin kasaamista hyvin lyhyiden ja yksinkertaisten esimerkkien avulla. Tavoitteena oli pohjustaa aihetta helposti lähestyttävin esimerkein vaikka todellisuus onkin monimutkaisempi. Genomia kasaaviin ohjelmistoihin liittyy monia piirteitä, joita tässä tutkielmassa ainoastaan sivuttiin lyhyesti. Yksi niistä on virheiden ja toistojen käsittely, joka on laaja valikoima monimutkaisia algoritmeja ja tärkeä osa laadukkaan lopputuloksen takaamiseksi. Lisäksi teemaan liittyy oleellisesti myös ohjelmointiteknisiä kysymyksiä, kuten optimointi ja rinnakkaisuuksien hallinta ajon aikana. Myös ohjelmistojen tuottamien tulosten laadun raportointi ja niiden syvempi vertailu jätettiin pois, koska rajaus pyrittiin pitämään tietorakenteessa sinänsä ja sen ilmentymisessä: de Bruijn graafin rakenteen yksityiskohdat eivät suoranaisesti vaikuta kasatun genomin laatuun vaan erot siinä ovat seurausta useamman muun tekijän yhteisvaikutuksesta.

Vaikka de Bruijn graafeihin perustuva biologisen datan järjestely on tuottanut hyviä tuloksia, tietorakenne sinänsä on vasta osa ratkaisua. Täydellisiä tuloksia voi saada vain asiallisten sekvensointitekniikoiden ja ohjelmistojen yhteistoiminnalla ja kumpikin ala kehittyy edelleen.

Viiteluettelo

Nicolaas Govert de Bruijn. "*A Combinatorial Problem*" (1946)

Bentley DR, Balasubramanian S, Swerdlow HP, et al. "Accurate Whole Human Genome Sequencing using Reversible Terminator Chemistry". *Nature.* 2008; 456 (7218): 53–59.

Compeau, Phillip E. C., Pavel A. Pevzner, and Glenn Tesler. "How to Apply De Bruijn Graphs to Genome Assembly." *Nature biotechnology* 29.11 (2011): 987-91. ProQuest. 10 Nov. 2019

Jackman, Shaun D et al. "ABySS 2.0: resource-efficient assembly of large genomes using a Bloom filter." *Genome Research* vol. 27 (2017): 768-777. doi:10.1101/gr.214346.116

Khan, Abdul Rafay et al. “A Comprehensive Study of De Novo Genome Assemblers: Current Challenges and Future Prospective.” *Evolutionary bioinformatics online* vol. 14. 20 Feb. 2018, doi 10.1177/1176934318758650

Adam Kozak, Tomasz Głowacki, Piotr Formanowicz. "On a generalized model of labeled graphs." *Discrete Applied Mathematics,* Volume 161, Sep. 2013, Pages 1818-1827

Eduardo Moreno. "De Bruijn sequences and De Bruijn graphs for a general language." *Information Processing Letters* Volume 96, Issue 6, 31 December 2005, Pages 214-219

Pevzner, P A et al. “An Eulerian path approach to DNA fragment assembly.” *Proceedings of the National Academy of Sciences of the United States of America* vol. 98,17 (2001): 9748-9753. doi:10.1073/pnas.171285098

Simpson, Jared T et al. “ABySS: a parallel assembler for short read sequence data.” *Genome research* vol. 19,6 (2009): 1117-23. doi:10.1101/gr.089532.108

Aída Falcón de Vargas. "*The Human Genome Project and its importance in clinical medicine.*" International Congress Series Volume 1237, July 2002, Pages 3-13

Zerbino, Daniel R, and Ewan Birney. “Velvet: algorithms for de novo short read assembly using de Bruijn graphs.” *Genome research* vol. 18,5 (2008): 821-9. doi:10.1101/gr.074492.107